1. Find the error in each of the following program segments. Assume the following declarations and statements:

```
int *zPtr; // zPtr will reference array z
void *sPtr = 0;
int number;
int z[ 5 ] = { 1, 2, 3, 4, 5 };
```

```
a. ++zPtr;
b. // use pointer to get first value of array
   number = zPtr;
c. // assign array element 2 (the value 3) to number
   number = *zPtr[ 2 ];
d. // print entire array z
   for ( int i = 0; i <= 5; i++ )
        cout << zPtr[ i ] << endl;
e. // assign the value pointed to by sPtr to number
   number = *sPtr;
f. ++z;
```

a. *Error:* zPtr has not been initialized.
Correction: Initialize zPtr with zPtr = z;

b. *Error:* The pointer is not dereferenced.
Correction: Change the statement to number = *zPtr;

c. *Error:* zPtr[2] is not a pointer and should not be dereferenced.
Correction: Change *zPtr[2] to zPtr[2].

d. *Error:* Referring to an array element outside the array bounds with pointer subscripting.
Correction: To prevent this, change the relational operator in the for statement to < or change the 5 to a 4

e. *Error:* Dereferencing a void pointer.
Correction: To dereference the void pointer, it must first be cast to an integer pointer. Change the statement to number = *static_cast< int * >(sPtr);

f. *Error:* Trying to modify an array name with pointer arithmetic.
Correction: Use a pointer variable instead of the array name to accomplish pointer arithmetic, or subscript the array name to refer to a specific element.

2. Write a function that **finds the smallest element** in an array of integers using the following header:

double min(double array[], int size)

Write a test program that prompts the user to enter ten numbers, invokes this function, and displays the minimum value. Here is the sample run of the program:

<Output>

Enter ten numbers: 1.9 2.5 3.7 2 1.5 6 3 4 5 2

The minimum number is: 1.5

<End Output>

```
#include <iostream>
#include <cmath>
using namespace std;

double min(double list[], int size)
{
  double min = list[0];

  for (int i = 1; i < size; i++)
    if (min > list[i])
      min = list[i];

  return min;
}

int main()
{
  const int SIZE = 10;
  double numbers[SIZE];
  cout << "Enter ten numbers: ";
  for (int i = 0; i < SIZE; i++)
  {
    cin >> numbers[i];
  }

  cout << "The minimum number is " << min(numbers, SIZE) << endl;

  return 0;
}
```

3. Deitel 7.11

(Bubble Sort) In the bubble sort algorithm, smaller values gradually "bubble" their way upward to the top of the array like air bubbles rising in water, while the larger values sink to the bottom. The bubble sort makes several passes through the array. On each pass, successive pairs of elements are compared. If a pair is in increasing order (or the values are identical), we leave the values as they are. If a pair is in decreasing order, their values are swapped in the array. Write a program that sorts an array of 10 integers using bubble sort.

```
// Exercise 6.11 Solution: ex06_11.cpp
// This program sorts an array's values into ascending order.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    const int arraySize = 10; // size of array a
    int a[ arraySize ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
    int hold; // temporary location used to swap array elements

    cout << "Data items in original order\n";

// output original array
    for ( int i = 0; i < arraySize; i++ )
        cout << setw( 4 ) << a[ i ];

// bubble sort
// loop to control number of passes
    for ( int pass = 0; pass < arraySize - 1; pass++ )
    {
// loop to control number of comparisons per pass
        for ( int j = 0; j < arraySize-1; j++ )
        {
// compare side-by-side elements and swap them if
// first element is greater than second element
            if ( a[ j ] > a[ j + 1 ] )
            {
                hold = a[ j ];
                a[ j ] = a[ j + 1 ];
                a[ j + 1 ] = hold;
            } // end if
        } // end for
    } // end for

    cout << "\nData items in ascending order\n";

// output sorted array
    for ( int k = 0; k < arraySize; k++ )
        cout << setw( 4 ) << a[ k ];

    cout << endl;
} // end main
```

4. What is the output of the program below.

<table>
<tr>
<td>

```
#include <iostream>
using namespace std;

void deneme1 (int *p, int *&q)
{
    *p = 100;
    p = p + 2;
    *p = *q;
    *q = *(p+1);
    cout << "p=" << *p << " q=" << *q << "\n";
}
bool deneme2 (int x[], int boyut)
{
    int *p = x;
    int *q;
    q = p+2;
    for (int i=1; i<boyut; i++)
    {
        x[i] += x[i-1];
        cout << x[i] << " ";
    }
    cout << "\n";
    deneme1 (p, q);
    cout<<"Again"<<endl;
    cout << "p=" << *p << " q=" << *q << "\n";
    return (*p == *q);
}
int main ( )
{
    int a[5] = {2,4,6,8,10};
    if (deneme2(a,5))
        cout << "correct\n";
    else
        cout << "nor correct\n";
    for (int i=0; i<5; i++)
        cout << a[i] << " ";
    return 0;
}
```

</td>
<td>
6 12 20 30
p=20 q=20
Again
p=100 q=20
not correct
100 6 20 20 30
</td>
</tr>
</table>